Analisa Kinerja Jaringan Peer-to-Peer (P2P) *BitTorrent*

Wan Delkey Vernata*, Febrizal**
*Alumni Teknik Elektro Universitas Riau, **Jurusan Teknik Elektro Universitas Riau
Kampus Bina Widya Km 12,5 Simpang Baru Panam Pekanbaru, 28293
Jurusan Teknik Elektro Universitas Riau
E-mail : wandelkey@live.com

ABSTRACT

Since its creation by Bram Cohen in 2001, BitTorrent has proven to provide the best way to file distribution among peer-to-peer system. Instead of establishing a single one-way stream of information like a client-server system, BitTorrent makes several connections to other clients that contain at least part of the desired information. This information is then simultaneously downloaded from and uploaded to the other clients in pieces. To design a close-to-real-world testing environment for BitTorrent experiment at the costs of a real hardware solution in this paper presents a BitTorrent performance analysis with virtualization. From results of the experiment suggested BitTorrent get faster distribution time compared to FTP with more requests and can be slower if have to deal with free riders clients. At the piece selection experiment, suggested BiTorrent get best distribution time at piece with numbers 1400.

Keyword: Peer-to-Peer, BitTorrent, Virtualisation, Performance Analisys

I. Pendahuluan

Secara umum untuk mendistribusikan file kepada klien adalah dengan menempatkan file tersebut pada sebuah server dan klien melakukan pengunduhan secara langsung dari server. Arsitektur jaringan seperti ini dinamakan sebagai klien-server, dimana server digunakan sebagai direktori dan melayani permintaan lebih dari satu klien pada waktu yang sama.

Permasalahan dalam arsitektur klien-server adalah adanya kemungkinan server tidak mampu untuk mengirim file besar dalam melayani jumlah *audiens* yang besar. Jika server melakukanya maka server akan mengalami peningkatan waktu untuk distribusi sehingga kinerja server menjadi tidak dapat diterima. Bisa disimpulkan semakin besar ukuran file yang diminta dan semakin besar jumlah *audiens* yang harus dilayani maka semakin buruk pula masalah kinerja yang akan dialami server. Hal ini juga disebut sebagai pajak popularitas file yang berada pada server.

Solusi yang dilakukan untuk masalah ini adalah dengan membeli server yang lebih kuat. Tetapi, server yang lebih kuat juga akan memiliki kapasitas yang terbatas dan memiliki resiko yang sama disaat beban puncak terjadi. Solusi lain untuk menangani masalah ini adalah dengan mengunakan arsitektur *peer-to-peer* (P2P), memanfaatkan gabungan dan pengolahan kekuatan jaringan klien individu untuk mengurangi beban pada server. Hal ini telah dibuktikan dengan banyak munculnya protokol P2P seperti *BitTorrent*, *Kaaza*, dan *Gnutella*.

Analisa kinerja jaringan P2P protokol *Bittorrent* dilakukan pada sebuah jaringan yang dibuat secara virtual. Pengunaan metode virtual digunakan sebagai solusi yang berkaitan dengan jumlah *peer* yang dibutuhkan pada jaringan P2P. Dengan menggunakan metode virtual ini akan lebih hemat biaya dan mengurangi jumlah penggunaan *hardware*.

Perancangan jaringan virtual dilakukan pada aplikasi *VMware Workstation*. Setiap *peer* akan diwakilkan oleh sebuah *Virtual Machine* (VMs) atau "Perangkat Virtual" yang

diciptakan dalam sebuah *interface* perangkat lunak *Workstation*. Metode virtual ini berbeda dengan simulasi P2P yang ada, karena pada penggunaan metode ini proses simulasi dilakukan lebih dekat-ke-dunia nyata

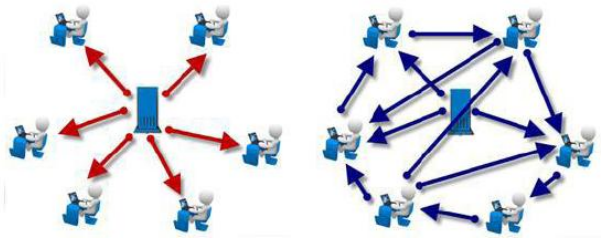
II. Dasar Teori

2.1. BitTorrent

BitTorrent adalah protokol yang paling populer digunakan untuk mendistribusikan file secara P2P. Survey *Sandvine* tahun 2014 menyatakan 14,71% trafiknya di benua Eropa adalah pengunaan *BitTorrent*. Hal ini membuat *BitTorrent* berada pada posisi ketiga dan *Youtube* menduduki posisi pertama dengan 17,38%.

BitTorrent umumnya digunakan oleh organisasi maupun individual saat harus mendistribusikan file-file yang relatif besar. Prinsip P2P pada *BitTorrent* akan mengurangi beban unggah dari pihak penyedia dalam melayani jumlah permintaan yang banyak.

Pada mulanya protokol *BitTorrent* diciptakan oleh Bram Cohen pada bulan April 2001 dan dirilis pada Juli 2001. Protokol ini sekarang dikembangkan oleh *Cohen's company BitTorrent*, *Inc*. Tidak seperti protokol P2P lainnya *(KaaZa dan Gnutella)*, ada banyak program implementasi lain dari klien *BitTorrent*, hingga saat ini sudah banyak implementasi lain dari klien *BitTorrent* yang ada, misalnya *Vuze* (dahlunya *Azureus*).



Gambar 2.1 Perbedaan Normal *Network* dan *BitTorrent Network*

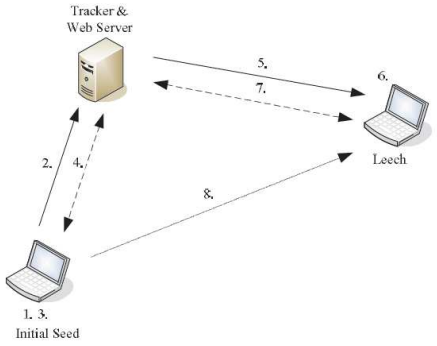
Pada gambar 2.1, sebuah normal *network* atau klien-server proses pengunduhan dilakukan setiap klien dengan melakuan permintaan terhadap file yang berada pada sebuah tunggal. Pada *BitTorrent* server digunakan sebagai pengiklan dan pelacak terhadap file yang klien inginkan. Proses distribusi file pada *BitTorrent* ditanggung setiap klien yang berada dalam jaringan.

2.2. Proses Pengiriman File BitTorrent

Proses menerima file dalam *BitTorrent* disebut “Unduh” dan proses memberikan file kepada *peer* lain disebut "Unggah." Setiap *peer* yang terlibat dalam kegiatan ini dikenal sebagai "pengunggah" dan "pengunduh".

Banyak proses yang tidak terlihat oleh seorang *end-user* pada saat mengunduh sebuah file menggunakan *BitTorrent*. Setiap pengguna hanya menemukan sebuah file (dot)torrent di Internet, yang disajikan dengan perintah "*Save As*" pada kotak dialog seperti yang digunakan dalam standar mengunduh file pada klien-server.

Struktur pengirimin file pada *BitTorrent* dijelaskan pada gambar berikut ini:



Gambar 2.2 Sturuktur Pengiriman File pada *BitTorrent*

Gambar 2.2 Sturuktur pengiriman file pada *BitTorrent,* Adapun penjelasan mengenai struktur diatas diejelaskan sebagai berikut:

1. Membuat file (dot)torrent mengunakan klien
2. Mengunggah file(dot)torrent pada *web server*

3. Buka file (dot)torrent dan mulai melakukan pengunggahan
4. Mengumumkan status kepemilikan pada *tracker* dan menungu balasan daftar *peer* dari *tracker*
5. Mengunduh file (dot)torrent dari *web server*
6. Buka file (dot)torrent dan mulai melakukan pengunduhan
7. Mengumumkan status kepemilikan pada *tracker* dan menungu balasan daftar *peer* dari *tracker*
8. Pertukaran potongan dilakukan antara klien

Selama proses pengunduhan, *peer* kembali menghubungi *tracker* pada interval waktu tertentu untuk memberikan pembaharuan kemajuan (butir 4 dan 7). Laporan kemajuan ini mencakup rincian berapa banyak data yang diunduh dan diunggah oleh sebuah *peer*. Interval waktu pelaporan ditentukan oleh *tracker.*

2.2.1. Tracker

Tracker (pelacak) adalah alasan *BitTorrent* merupakan sistem P2P terpusat. *Tracker* adalah server yang bertugas untuk melacak (peran *discovery*) setiap *peer* yang mengunduh dan mengunggah sebuah file pada protokol *BitTorrent*. Setiap *peer* yang ingin mengunduh harus mulai dengan menghubungi *tracker* dan mengumumkan kepemilikan file tertentu. *Tracker* akan memberikan daftar *peer* yang sudah memiliki sebagian file atau seluruh file.

Setelah mendapat informasi pengunggah dari *tracker,* maka pengunduh yang baru saja bergabung akan mengirimkan permintaan *handshake* pada pengunggah. Permintaan ini harus dikirimkan sebelum pengunduh dapat meminta potongan dari sebuah file kepada pengunggah. Pengunggah juga menginformasikan pada pengunduh berapa banyak potongan file dimiliki. Pengunduh dapat menggunakan informasi tersebut untuk memilih potongan yang akan diminta.

2.2.2. File (dot)torrent

Pada sistem distribusi *BitTorrent* File (dot)torrent berisi *metadata*, yang berisi informasi alamat *tracker* dan informasi mengenai potongan yang dibutuhkan untuk mengambil sebuah file. Informasi ini dibutuhkan oleh klien untuk *BitTorrent* untuk melakukan unduhan terhadap sebuah file dan verifikasi integritas file yang diunduh.

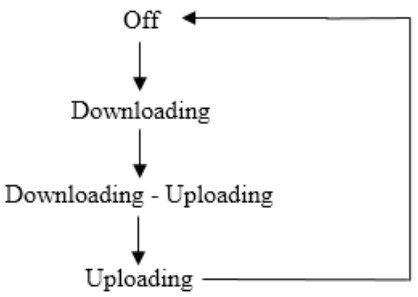
2.2.3. Klien-Klien BitTorrent

Klien-klien *BitTorrent* adalah perangkat lunak yang dibutuhkan sebuah *peer* untuk mengunduh sebuah file pada protokol *BitTorrent*. Seluruh proses menghubungi *tracker* dan klien-klienya lainnya semua ditangani dan diatur oleh masing-masing individu perangkat lunak klien *BitTorrent*

2.2.4. Istilah Peer Pada BitTorrent

Setiap *peer* yang mengunggah file lengkap dikenal sebagai "*seeder*" sementara setiap *peer* yang masih dalam proses unduh dan memiliki salinan parsial dari file tersebut dikenal sebagai "*leecher*". Harus ada setidaknya satu *seeder* hadir untuk memperkenalkan file ke dalam sistem. Istilah *seeder* adalah *peer* pertama yang beriklan di *tracker* dan awalnya menjadi satu-satunya pengunggah. Semua pengunduh harus mulai mendapatkan file dari *seeder.*

Siklus dari *peer BitTorrent* dapat diringkas seperti yang ditunjukkan pada gambar 2.2.



Gambar 2.2. Siklus Dari Sebuah *BitTorrent*

Pada kenyataanya, hanya beberapa *peer* yang beroperasi sesuai dengan siklus, sebagian

peer yang hanya akan mengunduh tetapi tidak melakukan pengunggahan sama sekali, *peer* ini disebut "*Free Riders*" atau "*A Hit and Runner*". *Free riders* menyebabkan masalah pada dunia P2P karena mereka mengkonsumsi sumber daya tetapi tidak memberikan apa-apa kepada *peer* lainnya.

Upaya *BitTorrent* untuk mengatasi masalah ini adalah dengan menggunakan algoritma *chocking*. *Chocking* adalah penolakan sementara untuk mengunggah potongan file kepada pengunduh tertentu. Algoritma Ini memungkinkan sebuah pengunggah tidak menerima terlalu banyak koneksi dari *peer* yang mengunduh. Tujuan utama dari algoritma *chocking* adalah untuk memastikan *peer* tidak memberikan banyak kepada sistem agar tidak menerima banyak imbalan. Algoritma ini juga dikenal sebagai *tit-for-tat*.

2.2.5. Potongan File

Besar file yang didistribusikan menggunakan *BitTorrent* dibagi menjadi beberapa potongan. Setiap kali potongan diterima, potongan tersebut akan diperiksa menggunakan algoritma SHA1 untuk memverifikasi kesamaan data. *BitTorrent* bergantung pada penyebaran potongan di dalam *swarm*. Dengan adanya algoritma *tit-for-tat* potongan pertama umumnya yang paling sulit untuk didapatkan. Semakin kecil potongan, maka semakin mudah bagi setiap *peer* untuk mendapatkannya.

Dalam memilih potongan yang ingin selanjutnya untuk diunduh, *BitTorrent* mengunakan algoritma "*the rarest-first*". Dengan algoritma ini setiap klien akan memilih potongan yang paling langka pertama dan acak jika semua potongan ketersediaan.

Setiap kali potongan penuh diterima klien akan mengirimkan pesan "*Have*" kepada setiap *peer*. Semakin banyak potongan maka semakin banyak *bandwidth* yang digunakan untuk protokol data.

2.2.6. Endgame Mode

Menjelang akhir unduhan ketika *peer* pengunduh telah menerima sebagian besar potongan file *peer* tersebut dapat masuk pada *endgame mode*. Pada *endgame mode* pengunduh mengirimkan permintaan untuk potongan yang sama kepada setiap *peer*. Setelah potongan dibalas oleh seorang *peer*, pengunduh kemudian mengirimkan pesan pembatalan kepada *peer* lainnya.

Sementara *endgame mode* membuat *peer* mengirimkan permintaan untuk potongan yang sama dapat mempercepat proses unduhan. Strategi ini tidak begitu efisien, dalam beberapa kasus beberapa besar jumlah permintaan yang dikirimkan memungkinkan untuk seseorang pengunggah sudah mengirimkan potongan sebelum permintaan pembatalan tiba. Hasilnya rangkap salinan yang tidak diinginkan dari potongan yang diterima oleh pengunduh terbuang percuma. Sehingga dapat disimpulkan bahwa *peer* tidak perlu masuk kedalam *endgame mode* terlalu dini. Tidak ada aturan tetap untuk seorang pengunduh untuk memasuki *endgame mode*, hal ini ditentukan oleh klien *BitTorrent* itu sendiri.

III. Metode Penelitian

Dalam menciptakan atau pengadaan dan konfigurasi perangkat keras, perangkat lunak, dan antarmuka, bisa menjadi mahal dan memakan waktu. Dengan metode virtualisasi pengujian dilakukan dalam lingkungan yang lebih hemat biaya. Perancangan jaringan virtual dilakukan pada aplikasi *VMware Workstation*. Setiap *peer* akan diwakilkan oleh sebuah *Virtual Machine* (VMs) atau "Perangkat Virtual". Metode virtual ini berbeda dengan simulasi P2P yang ada, karena pada menggunakan metode ini proses simulasi dekat-ke-dunia nyata.

3.1. Implementasi Sistem

3.1.1. Virtualisasi mengunakan VMware Workstation

Vmware Workstation adalah adalah perangkat lunak yang umum digunakan untuk menciptakan sebuah perangkat virtual.

:

Guest OS	Guest OS	Guest OS
Hardware	Hardware	Hardware
Workstation		
Host OS		
Hardware		

Gambar 3.1 Arsitektur Virtualisasi

Pada Gambar 3.1, arsitektur pemasangan perangkat lunak *Workstation* membutuhkan sebuah sistem operasi dasar. Kelebihan pada arsitektur ini memungkinkan penggunaan perangkat virtualisasi yang lebih fleksibel. Kekurangan pada arsitektur ini adalah dapat mengurangi kinerja dari *host*, karna harus berbagi sumber daya pada setiap perangkat virtual.

3.1.1.1. Perangkat Virtual

Jumlah perangkat virtual yang dapat dijalankan bersamaan pada *Workstation* tergantung pada kapasitas penyimpanan dan memori yang dimiliki *host* dan spesifikasi dari perangkat virtual. Semakin besar spesifikasi yang dimiliki *host* maka semakin besar jumlah perangkat virtual yang dapat dijalankan n. Berdasarkan hal tersebut spesifikasi *host* digunakan dan perangkat virtual yang diciptakan pada skripsi ini dijelaskan sebagai berikut:

Spesifikasi dari *host* dan perangkat virtual yang diciptakan, dijelaskan sebagai berikut:

1. **Spesifikasi komputer sebagai *host* yang digunakan:**
   - *Intel® Core™ i*5-2450M CPU @ 2.50GHz (4CPU,s)
   - *Windows 7 Home Premium 64-bit* (6,1 *build* 7601)
   - 4096 MB RAM
   - 640 GB HDD

2. **Spesifikasi perangkat virtual:**
   - *Puppy Linux, Slacko Puppy* 6.3
   - 128 MB RAM
   - 8 GB HDD

Pada dasarnya sistem operasi *Puppy Linux* hanya membutuhkan 32 MB memori untuk minimal. Namun, untuk menjalakan klien *Bittorent* dan mendapatkan kinerja perangkat virtual yang baik diputuskan untuk mengalokasikan 128 MB.

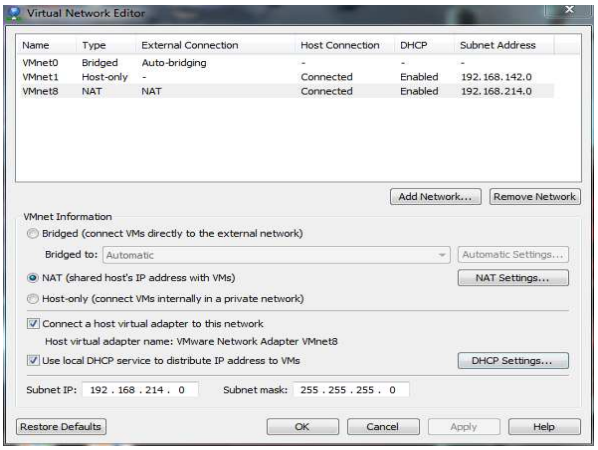
3.1.1.2. Nat Pada Virtual Network

Secara *default* setiap perangkat virtual akan terhubung kepada adapter *Virtual Network* yang disediakan infrastruktur *Workstation*, dijelaskan seperti berikut:

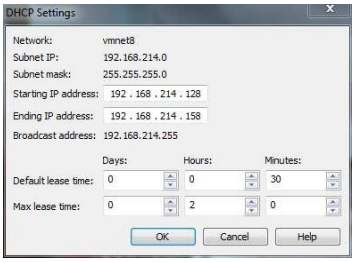
1. ***VmNet0 (Bridged)***
2. ***VmNet1 (Host only)***
3. ***VmNet8 (NAT)***

Pada NAT atau *Network Address Translation*, setiap perangkat virtual akan menerima alamat IP dari *Workstation* yang ditentukan server DHCP. Pengaturan IP *default* digunakan untuk jaringan NAT adalah kelas C (atau bisa diatur sedemikian rupa, seperti pada gambar 3.2 dan gambar 3.3).

Pengaturan NAT dan DHCP pada *virtual network* ditunjukan oleh gambar berikut:



Gambar 3.2. Pengaturan NAT Adapter *VmNet8*



Gambar 3.3. Pengaturan DHCP

Gambar 3.3 adalah pengaturan NAT pada Virtual Network 8, setiap perangkat virtual akan terhubung pada adapter ini. Kelebihan utama NAT adalah menyediakan cara yang transparan dan mudah untuk mengkonfigurasi setiap perangkat virtual untuk mendapatkan akses pada jaringan.

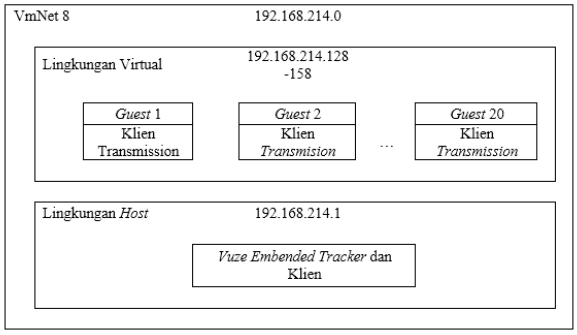
Gambar 3.2 adalah pengaturan server DHCP (*Dinamic Host Configuration Protocol*). Server DHCP bertugas memberi IP address kepada setiap perangkat virtual yang menginginkan akses pada jaringan. Pemberian IP dilakukan secara acak sesuai dengan rentang yang ditentukan.

3.1.2. Klien BitTorrent Dan Tracker

Dalam melakukan pengujian skripsi ini akan menggunakan klien-klien *BitTorrrent* yang tersedia pada Internet. Klien-klien ini akan mengambil peran dalam distribusi file pada protokol *BitTorrent*. Berdasar fitur dan fungsi klien untuk pengujian pada skripsi ini akan mengunakan klien-klien berikut:

- ***Vuze*** sebagai *tracker* dan *klien* pengunggah pada lingkungan host
- ***Transmission*** sebagai klien-klien pengunduh pada lingkungan virtual

Struktur dan pengaturan klien dan *tracker* dijelaskan pada gambar berikut:



Gambar 3.4 Pengaturan Klien *BitTorrent* dan *Tracker*

Pada gambar 3.4 peran pengungah akan dilakukan oleh klien Vuze pada lingkungan *host*. Fitur *tracker* pada klien *Vuze* memungkinkan klien ini untuk meng-*host* sebuah file (Dot)torrent dan mempublikasikan file tersebut pada *web server*-nya. Peran pengunduh akan dilakukan oleh klien *Transmission* di setiap perangkat virtual. Pada *Transmission* memiliki fitur *Web Interface* yang memungkinkan klien ini untuk dipantau dan dikontrol dari lingkungan *host*.

3.3 Pandangan Keseluruhan

Sebelum melakukan analisa kinerja terhadap protokol *BitTorrent* bagian ini akan menjelaskan sistem pengujian pengujian yang akan dilakukan sebagai berikut:

1. Didalam sistem yang telah dibuat jumlah klien maksimal adalah 21 klien (1 pada *host* klien *Vuze* dan 20 pada perangkat virtual klien *Transmission*).

2. Setiap klien akan melakukan simulasi unduh-unggah file pada jaringan untuk menghasilkan kinerja dari protokol *BitTorrent.* Langkah-langkah pengujian secara garis besar dijelakan sebagai berikut:

   - Mulai *tracker*,
   - Buat file *(Dot)torrent.*
   - Mulai klien *leech* dan melakukan *leech* terhadap file *(Dot)torrent*
   - Mulai klien *seed* dan melakukan *seed* terhadap file *(Dot)torrent*.

Berdasarkan alokasi *bandwidth* setiap klien di bagi menjadi dua, yaitu:

- *High bandwidth* klien(*upload speed* = 256, *download speed* = 512, kBps)
- *Low bandwidth* klien (*upload speed* = 32, *download speed* = 64, kBps).

3.4 Analisa Kinerja

Kinerja *BitTorrent* akan dianalisa berdasarkan waktu yang dibutuhkan oleh protokol saat mendistribusikan sebuah file. Waktu distribusi diambil pada *logging* klien *Vuze* sebagai klien pengunggah. Bagian-bagian analisa terhadap kinerja yang akan dilakukan adalah:

1. Kinerja protokol protokol *BitTorrent*
2. Pengaruh bertambahnya permintaaan terhadap waktu distribusi.
3. Pengaruh klien yang tidak berbagi *free riders* yang berbagi terhadap waktu distribusi.
4. Pengaruh ukuran dan jumlah potongan terhadap waktu pertukaran potongan dan waktu distribusi.

3.4.1. Skenario pengujian

Adapun skenario pengujian yang akan dilakukan seperti yang dijelaskan pada analisa kinerja adalah sebagai berikut:

3.4.1.1. Monitoring Kinerja *BitTorrent*

Pada skenario pengujian ini menggunakan 21 klien untuk mendistribuskan file berukuran 100 MB. Agar mendapatkan kinerja untuk dianalisa, setiap klien pengunduh dibagi kedalam 2 grup, yaitu 10 klien berstatus klien *high bandwidth* dan 10 klien berstatus *low bandwidth*. Klien *Vuze* sebagai pengunggah berstatus klien *high bandwidth*.

3.4.1.2. Pengaruh Bertambahnya Permintaan

Pada skenario pengujian ini menggunakan klien-klien *high bandwidth* untuk mendisitribusikan file berukuran 10 MB. Pengukuran dilakukan terhadap waktu distribusi saat menghadapi jumlah permintaan file sebesar 1, 5, 10, 15, dan 20. Pada pengujian ini juga dilakukan perbandingan dengan klien-server protokol FTP menggunakan server *FileZilla*.

3.4.1.3. Pengaruh Free Riders

Pada skenario pengujian ini menggunakan 21 klien *high bandwidth* untuk mendisitribusikan file berukuran 10 MB. Pengukuran dilakukan terhadap waktu distribusi saat menghadapi klien dengan prilaku *free riders* sebesar 1, 5, 10, 15 dan 20.

3.4.1.4. Pengaruh Potongan

Pada skenario pengujian ini mengunakan 3 klien *high bandwidth* dalam mendistribusikan file sebesar 350 MB. Pengukuran dilakukan pada waktu distribusi dan *swarm* pada ukuran potongan 64 kB, 256, kB 512, kB, 1 MB dan 2 MB.

IV. Hasil dan Pembahasan

4.1. Analisa Kinerja *BitTorrent*

Pada skenario pengujian ini dilakukan monitoring/pengamatan terhadap sudut

pandang klien dan perubahan kecepatan unggah saat distribusi berlangsung.

Dari hasil pengukuran waktu total pendistribusian adalah 1592 detik untuk setiap klien menyelesaikan unduhanya, dan 695 detik untuk klien *high bandwidth* menyelesaikan unduhanya.

Sudut pandang klien saat awal distribusi berlangsung dapat dilihat pada gambar 4.1. dan 4.2.

IP	Client	T	Pieces	%	Down Speed	Up Speed	State
192.168.214.148	Transmission 2.60	L		20.6%	32 B/s	0 B/s	Fully established
192.168.214.139	Transmission 2.60	L		9.2%	9 B/s	1 B/s	Fully established
192.168.214.149	Transmission 2.60	L		15.8%	30 B/s	1 B/s	Fully established
192.168.214.133	Transmission 2.60	L		4.5%	6 B/s	1 B/s	Fully established
192.168.214.157	Transmission 2.60	L		10.0%	21 B/s	1 B/s	Fully established
192.168.214.153	Transmission 2.60	L		19.8%	24 B/s	1 B/s	Fully established
192.168.214.155	Transmission 2.60	L		15.6%	18 B/s	1 B/s	Fully established
192.168.214.154	Transmission 2.60	L		20.1%	9 B/s	1 B/s	Fully established
192.168.214.135	Transmission 2.60	L		6.1%	23 B/s	3.2 kB/s	Fully established
192.168.214.138	Transmission 2.60	L		9.1%	22 B/s	3.2 kB/s	Fully established
192.168.214.136	Transmission 2.60	L		6.6%	14 B/s	6.4 kB/s	Fully established
192.168.214.142	Transmission 2.60	L		7.3%	24 B/s	7.0 kB/s	Fully established
192.168.214.151	Transmission 2.60	L		4.2%	52 B/s	7.7 kB/s	Fully established
192.168.214.137	Transmission 2.60	L		6.8%	25 B/s	9.6 kB/s	Fully established
192.168.214.152	Transmission 2.60	L		9.3%	51 B/s	16.7 kB/s	Fully established
192.168.214.143	Transmission 2.60	L		7.0%	46 B/s	19.0 kB/s	Fully established
192.168.214.141	Transmission 2.60	L		11.5%	39 B/s	28.9 kB/s	Fully established
192.168.214.150	Transmission 2.60	R		21.1%	43 B/s	31.4 kB/s	Fully established
192.168.214.140	Transmission 2.60	L		4.5%	46 B/s	32.0 kB/s	Fully established
192.168.214.156	Transmission 2.60	L		20.8%	59 B/s	84.0 kB/s	Fully established

Gambar 4.1 Sudut Pandang Dari Klien Pengunggah (*Vuze*) 192.168.214.1

Keterangan kolom pada gambar 4.1:

1. Ip: adalah alamat ip sebuah *peer*
2. *Client*: adalah jenis klien yang digunakan *peer*.
3. T: adalah status *local* dan *remote peer* (L jika *Vuze* yang mengirimkan permintaan *handshake*, R jika *peer* mengirimkan permintaan *handshake*)
4. *Pieces*: potongan yang dimiliki telah *peer*
5. *%*: Persentasi dari jumlah potongan
6. *Down Speed*: adalah kecepatan unduh yang diterima dari *peer*
7. *Up speed*: adalah kecepatan unggah yang diberikan kepada *peer*.
8. *State*: adalah status koneksi dengan *peer*.

Up	Down	%	Status	Address	Client
		100%	DI	192.168.214.1	Vuze 5.7.1.0
	1 kB/s	2%	TUKEHI	192.168.214.133	Transmission 2.60
6 kB/s		3%	TUKEHI	192.168.214.135	Transmission 2.60
8 kB/s	1 kB/s	3%	TDUEHI	192.168.214.136	Transmission 2.60
		1%	TDUEHI	192.168.214.137	Transmission 2.60
		5%	TDUEH	192.168.214.138	Transmission 2.60
		1%	TDuEHI	192.168.214.139	Transmission 2.60
2 kB/s	5 kB/s	2%	TDUEH	192.168.214.140	Transmission 2.60
		0%	TUKEHI	192.168.214.141	Transmission 2.60
		3%	TDUEHI	192.168.214.142	Transmission 2.60
		3%	TDUEHI	192.168.214.143	Transmission 2.60
	24 kB/s	13%	TD?E	192.168.214.148	Transmission 2.60
1 kB/s	8 kB/s	8%	TODUE	192.168.214.149	Transmission 2.60
	39 kB/s	12%	TDUE	192.168.214.150	Transmission 2.60
		0%	TUKEXI	192.168.214.151	Transmission 2.60
		3%	TE	192.168.214.152	Transmission 2.60
	38 kB/s	13%	TDUEI	192.168.214.154	Transmission 2.60
		5%	TDUEH	192.168.214.155	Transmission 2.60
	52 kB/s	11%	TDUEH	192.168.214.156	Transmission 2.60
		1%	TDuEH	192.168.214.157	Transmission 2.60

Gambar 4.2 Sudut Pandang Dari Klien Pengunduh (*Transmission*) 192.168.214.153.

Keterangan kolom pada gambar 4.2:

1. Up: adalah kecepatan unggah yang diberikan kepada *peer*.
2. Down: adalah kecepatan unduh yang diterima dari *peer*
3. *%*: Persentasi dari jumlah potongan yang telah dimiliki *peer*
4. *Status*: adalah status koneksi dengan *peer*.
5. Ip: adalah alamat ip dari *peer* yang terhubung pada klien
6. *Client*: adalah jenis klien yang digunakan *peer*.

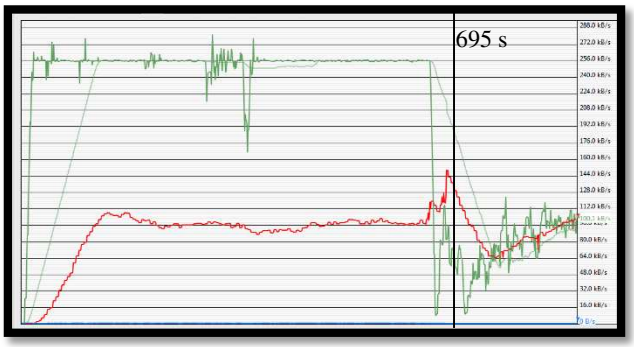
Gambar 4.1 dan 4.2 adalah sampel yang diambil pada saat distribusi dimulai. Setiap gambar mepresentasikan sudut pandang klien pada saat proses distribusi berlangsung.

Gambar 4.1 mempresentasikan sudut pandang klien *Vuze* sebagai pengunggah. Sebagai klien pengunggah pertama klien ini telah memiliki 100% dari potongan file yang ingin didistribusikan (dilihat pada gambar 4.2 baris pertama). Pada kolom *Up speed* dari klien ini adalah proses pengunggahan yang dilakukan klien ini kepada masing-masing klien *Transmission* sebagai pengunduh.

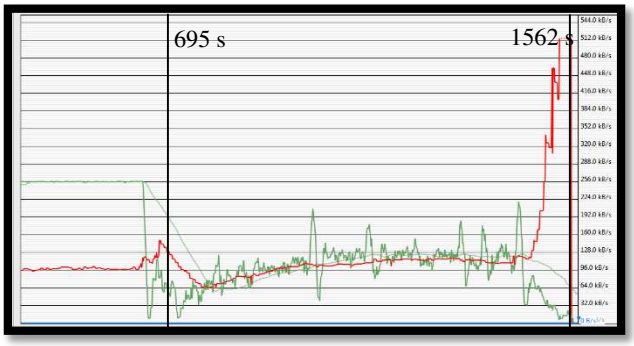
Gambar 4.2 mempresentasikan sudut pandang salah satu klien *Transmission* sebagai pengunduh. Sebagai klien pengunduh klien ini harus meminta potongan dari klien pengunggah dan klien pengunduh lainya. Selain mengunduh

klien ini juga akan mengunggah potongan yang dimilikinya kepada klien yang belum memiliki potongan yang sama (siklus *downloading-uploadin*g). Pada kolom Up dan Down adalah proses ungah dan unduh yang dilakukan klien ini kepada klien lainya. Proses unggah dan unduh ini juga dapat dilihat pada status D dan U pada kolom *State*, yang berarti "*Downloading from this peer*" dan "*Uploading to peer*".

Sementara mengamati pada sudut pandang klien pengungah dan pengunduh, pengamatan juga dilakukan terhadap perubahan kecepatan ungah dan *swarm.* Perubahan kecepatan unggah dan *swarm* pada klien *Vuze* dapat dilihat pada gambar berikut:



Gambar 4.3 Perubahan Kecepatan Pada Klien *Vuze* – Fase awal.



Gambar 4.4 Perubahan Kecepatan Pada Klien *Vuze* – Fase Akhir

Keterangan Gambar 4.3 dan 4.4 :

1. Kurva hijau: Kecepatan unggah.
2. Kurva merah: Kecepatan *swarm*
3. 2 garis hitam vertikal menandakan waktu 695 dan 1562 detik

Gambar 4.3 dan 4.5 mempresentasikan perubahan kecepatan unggah dan *swarm* yang dihasilkan pada klien *Vuze* sebagai pengunggah. Setiap gambar mempresentasikan waktu 15 menit pengamatan perubahan kecepatan.

Pada gambar 4.3 dapat dilihat kecepatan unggah klien ini berada stabil pada garis alokasi *bandwidth* unggah maksimal 256 kB hingga mendekati garis 695 detik. Penurunan kecepatan saat mendekati garis 695 disebabkan satu persatu klien pengunduh *high bandiwidth* telah menyelesaikan unduhanya. Sebagai klien pengunggah *high bandwitdh* penurununan kecepatan unggah ini juga dapat diartikan sebagai penurunan beban. Klien pengunduh yang telah menyelesaikan unduhanya akan menghapus klien pengunggah dari daftarnya dan hanya melakukan unggah terhadap klien yang belum menyelesaikan unduhanya (siklus *uploading*).

Pada gambar 4.3 juga dapat dilihat perubahan kecepatan *swarm.* Munculnya kecepatan *swarm* ini menandakan waktu dimana klien pengunduh mulai memiliki sebuah potongan dan mengunggah potongan tersebut terhadap klien pengunduh lainya (siklus *downloading-uploading*).

Pada gambar 4.3 dan 4.4 dapat dilihat tejadi dua kali peningkatan kecepatan *swarm* saat mendekati 695 detik dan 1562 detik. Peningkatan kecepatan *swarm* ini disebabkan oleh klien pengunduh yang hampir menyelesaikan unduhannya masuk pada *endgame mode*. Satu-satunya hasil yang tidak diduga dari pengamatan kecepatan *swarm* saat memasuki *endgame mode* adalah peningkatan kecepatan yang melebihi dari alokasi *bandwidth* yang diberikan kepada setiap klien. Pada hal ini *TransmissionBt* menjelaskan mengenai adanya beberapa *bug* pada kliennya yang menyebabkan korupsi data saat memasuki *endgame mode*.

3.2. Analisa Pengaruh Bertambahnya Permintaan

Pada skenario pengujian ini akan menyelidiki pengaruh bertambahnya permintaan sebuah file terhadap waktu distribusi protokol *BitTorrent* dan FTP. Hasil pengukuran pengaruh bertambahnya klien yang melakukan permintaan file 10 MB protokol *BitTorrent* dan FTP dapat dilihat pada tabel 4.1 dan 4.2.

Tabel 4.1 Pengaruh Bertambahnya Permintaan Pada *BitTorrent*

Jumah Permintaaan	Waktu Distribusi (s)
1	43.7
5	57.2
10	72.0
15	103.7
20	127.3

Tabel 4.1 Pengaruh Bertambahnya Permintaan Pada FTP

Jumlah Permintaan	Waktu Distribusi (s)
1	43.2
5	218.6
10	430.7
15	652.3
20	876.9

Pada tabel 4.1. dan 4.2 dapat dilihat hasil pengukuran kinerja *BitTorrent* dan FTP. Dari hasil yang didapat pada saat menghadapi 1 permintaan kinerja *BitTorrent* lebih lambat dibandingkan FTP. Namun, pada jumlah permintaan yang banyak dapat dilihat *BitTorrent* lebih cepat dibandingkan dengan FTP. Perbedaan waktu yang terbesar dapat dilihat pada pengujian 20 permintaan, dimana pada pengujian ini dapat dikatatakan dengan mengunakan *BitTorrent* lebih hemat waktu sebesar 749.6 detik dibandingkan FTP.

Pada pengukuran FTP perubahan waktu yang meningkat tajam dan bergerak linear beriring dengan bertambahnya permintaan pada server. Seperti yang telah dijelaskan sebelumnya arsitektur klien-server, server digunakan sebagai direktori dan melayani permintaan lebih dari satu klien pada waktu yang sama. Kehadiran peminta (*audiens*) yang semakin banyak akan memperburuk kinerja server. Pada pengamatan pengujian FTP 5, 10, 15, dan 20 setiap pengunduh mendapatkan kecepatan unduh yang dibagi dari 256 kB *bandwidth* unggah yang dimiliki oleh server.

Pada pengukuran *BitTorrent* perubahan waktu tidak meningkat setajam pada FTP. Seperti yang dijelaskan pada P2P semakin banyak *node-node* yang melakukan permintaan (*audiens*) kapasitas total dari sistem juga akan meningkat. Pada pengujian *BitTtorrent* setiap kehadiran pengunduh juga dapat dianalogikan sebagai kehadiran pengunggah. Kehadiran pengunduh yang memasuki siklus *downloading-uploading* maupun *uploading* pada klien *BitTorrent* membuat perubahan peningkatan waktu distribusi lebih sedikit dibandingkan FTP. Pada pengamatan pengujian 5, 10, 15, dan 20 permintaan *BitTorrent* setiap pengunduh saling bergantian mendapatkan kecepatan unduhan maksimal tergantung pada jumlah klien yang terhubung, kecepatan unduh yang ditawarkan, dan ketersediaan potongan didalam *swarm* pada waktu tertentu.

3.3. Analisa Pengaruh *Free Rider*

Pada skenario pengujian ini akan menyelidiki pengaruh bertambahnya prilaku *free rider* pada protokol *BitTorrent* saat menghadapi 20 permintaan. Hasil pengukuran pengaruh bertambahnya prilaku *free rider* pada protokol *BitTorrent* medistribusikan file 10 MB dapat dilihat pada tabel 4.3.

Tabel 4.3 Pengaruh Bertambahnya *Free Rider*

Jumlah *Free Rider*	Waktu Distribusi (s)
1	136.3
5	177.5
10	217.8
15	470.7
20	897.2

Pada Tabel 4.3 dapat dilihat pengaruh *free riders* terhadap waktu distribusi *BitTorrent*. Dari tabel ini jika dibandingkan dengan keaadan ideal *BitTorrent* dengan 20 permintaan (tabel 4.1), hadirnya klien *free rider* memperlambat kinerja yang dihasilkan. Semakin bertambah klien *free riders*, maka kinerja yang dihasilkan juga semakin lambat.

Perbedaan kinerja yang paling lama adalah pada saat 20 klien *free riders* yang melakukan permintaan didalam sistem. Pada pengujian 20 *free riders* dapat dianalogikan sebagai pengujian *BitTorrent* dengan arsitektur klien-server. Dari hasil pengukuran ini jika dibandingkan dengan 20 permintaan pada FTP (tabel 4.2) *BitTorrent* lebih lambat 20.3 detik. Pada pengamatan pengujian *free riders* ini ada 2 faktor yang mempengaruhi lambatnya kinerja protokol *BitTorrent*. Faktor pertama adalah pengiriman file yang dibagi menjadi beberapa potongan dan faktor kedua adalah aturan *tit-for-tat*. Pada faktor pertama setiap potongan yang dikirimkan memiliki nilai *delay,* hal ini dapat dilihat lebih jelas pada pengaruh potongan terhadap waktu distribusi pada pengukuran dibawah (tabel 4.4). Pada faktor kedua *tit-for-tat* yang mewajibkan sebuah *peer* untuk mengunggah berberapa data atau harus mendapatkan waktu pinalti sebelum dapat menerima potongan dari klien lainya. Pada beberapa kasus pengujian 15 dan 20 *free riders* ditemukan status "d" atau "*downloading from this peer if they let us*" pada klien *free riders* yang mengartikan klien ini sedang di-*chocke* oleh sebuah klien.

4.4 Analisa Pengaruh Potongan

Pada skenario pengujian ini akan menyelidiki pengaruh ukuran yang ditentukan pada sebuah file terhadap efisensi pertukaran potongan didalam sistem. Hasil pengukuran pengaruh ukuran potongan pada file berukuran 350 MB dapat dilihat pada tabel 4.4.

Tabel 4.4 Pengaruh Potongan Pada Waktu Swarm Dan Distribusi Klien

Ukuran Potongan	Jumlah Potongan	Waktu Swarm (s)	Waktu Distribusi (s)
64 kB	**5600**	3.8	1644.1
256 kB	**1400**	9.5	1565.2
512	**700**	16.2	1597.4
1 MB	**350**	22.2	1609.6
2 MB	**175**	35.7	1642.7

Pada tabel 4.4 dapat dilihat semakin kecil ukuran potongan maka semakin cepat dimulainya waktu *swarm*. Hal ini menyebabkan pertukaran potongan didalam sistem jadi lebih cepat dimulai. Namun, pada pengukuran ukuran potongan memiliki jumlah potongan yang lebih banyak dan menyebakan *delay* lebih banyak. Dari hasil pengukuran potongan yang paling kecil 64 kB ada 11200 potongan yang harus didistribusikan didalam sistem mendapatkan waktu distribusi paling lama dibandingkan ukuran potongan lainya. Pada pengamatan pengukuran potongan yang paling kecil ini lebih banyak waktu dihabiskan klien pengunduh untuk mengirimkan pesan permintaan (*Request*) dan kepemilikan (*Have*) dalam fungsi *logging Vuze.*

Pada pengukuran potongan 512 kB 1MB, dan 2 MB dapat dilihat waktu distribusi yang juga relatif lama dengan 64 kB. Pada pengamatan potongan besar lamanya waktu distribusi disebakan oleh pertukaran potongan kurang efisien didalam sistem. Dari pengukuran waktu *swarm* dapat dilihat pada ukuran potongan yang semakin besar klien pengunduh membutuhkan waktu yang semakin

lama untuk mengunduh sebuah potongan sebelum dapat mengunggah pada klien pengunduh lainya. Selain itu pada lamanya waktu distribusi potongan yang besar seperti pada kasus 1MB dan 2 MB, jumlah potongan terlalu sedikit dan menyebabkan sistem rentan menghadapi jeda yang disebabkan klien pengunduh tidak mengunggah jika memiliki potongan yang sama.

Hasil pengukuran potongan yang tercepat dapat dilihat pengukuran potongan 256 kB. Pada pengukuran ini jumlah potongan sebanyak 1400 potongan mendapatkan efisiensi yang terbaik. Hal yang sama juga ditemukan pada artikel *Vuze* mengenai pemilihan jumlah potongan agar berada 1000 hingga 1500 potongan.

V. Kesimpulan dan Saran

5.1 Kesimpulan

Penelitian yang disajikan dalam skripsi telah meneliti potensi P2P untuk mengatasi masalah kinerja dari sistem yang sepenuhnya tergantung pada kinerja server. Pada bab 4 menyajikan kinerja *BitTorrent*. Berdasarkan hal tersebut pada skripsi ini dapat dimuat menjadi beberapa kesimpulan yaitu:

1. Kinerja P2P lebih baik dibandingkan pada klien-server saat menghadapi jumlah permintaan yang lebih banyak dibandingkan klien server. Pada pengukuran 20 permintaan mengunakan *BitTorrent* lebih hemat waktu sebesar 749.6 detik dibandingkan FTP.
2. Pengaruh *free rider* atau klien yang tidak berbagi dalam sistem *BiTorrent* dapat memperburuk kinerja pada *BitTorrent*. Pada pengukuran 20 *BitTorrent free riders* kinerja *BitTorrent* yang dihasilkan 20.3 detik lebih lama dibandingkan 20 permintaan pada FTP.
3. Pemilihan potongan dengan jumlah 1400 yang mendapatkan durasi distribusi terbaik. Hal ini sesuai dengan saran *Vuze* jumlah potongan dari sebuah file agar berada pada 1000-1500 potongan untuk mendapatkan hasil pertukaraan yang optimal.

5.2 Saran

Adapun saran yang ingin diberikan berdasarkan metode dan hasil analisa dari skripsi ini adalah:

1. Memanfaatkan virtualisasi: Metode virtual yang diusulkan pada skripsi ini memungkinkan evaluasi kinerja *BitTorrent* dengan dalam sistem nyata yang lebih hemat biaya.
2. Memanfaatkan *BiTorrent*: Harapan dari dari penelitian ini agar lebih banyak organisasi maupun idividual yang ingin berbagi memanfaatkan protokol ini dalam mendistribusikan file mereka maupun file-file pihak ketiga yang sudah berlisensi.
3. Saran Penelitian: Penelitian terhadap *delay BitTorrent*. Penelitian yang sama juga dapat dilakukan untuk menganalisa *delay* dari protokol *BitTorrent* mengunakan *Network Analyzer* seperti *Wireshark* dan *Capsa*. Saran khusus mengenai penelitian ini agar mengunakan klien pengunduh yang berbeda (bukan *Transmission*) untuk menghindari korupsi data.

DAFTAR PUSTAKA

____. *BitTorrent.* http://www.bittorrent.com/index.html

____. *The BitTorrent Protocol.* http://www.bittorrent.org/beps/bep_0003.html

____. *The BitTorrent Specification.* http://wiki.theory.org/bittorrentspecification

____. *The VMware Workstation: Multiple Operating System.* http://www.vmware.com/product/workstation

____. *Torrent Piece Size*. https://wiki.vuze.com/w/torrent_piece_size

____. *Transmission Changes Log*.https://trac.transmissionbt.com/wiki/changes

D. Razvan. *Protocol Measurements and Improvements in Peer-to-Peer Systems University POLITECHNICA,* Bucharest, 2011, Hal: 29-86.

Hermawan Endah. *Analisa Cara Kerja dan Konfigurasi Torrent*, *Sekolah Tinggi Manajemen Informatika dan Komputer AMIKOM,* Yogyakarta, 2010, Hal: 26-27

John Colquhoun. *A BitTorrent-Based Peer-to-Peer Database Server*, *School of Computing Science Newcastle University,* United Kingdom, 2008, Hal: 11-16.

Lincoln Scully. *Network File Distribution with the BitTorrent Protocol, Saint Mary's University of Minnesota.Terrace Heights-Winona*. 2011

Sandvine. *Global Networks Phenomena Reports Waterloo*, Ontario Canada, April 2014.

Teemu Rautio. *Enhanced Peer Discovery For Multiaccess Peer-To-Peer Networks*, *Department of Electrical and Information Engineering University of OULU*, 2010, Hal 21-23.